TRIGGER (**g-2**): Raspberry Pi solution for beam testing

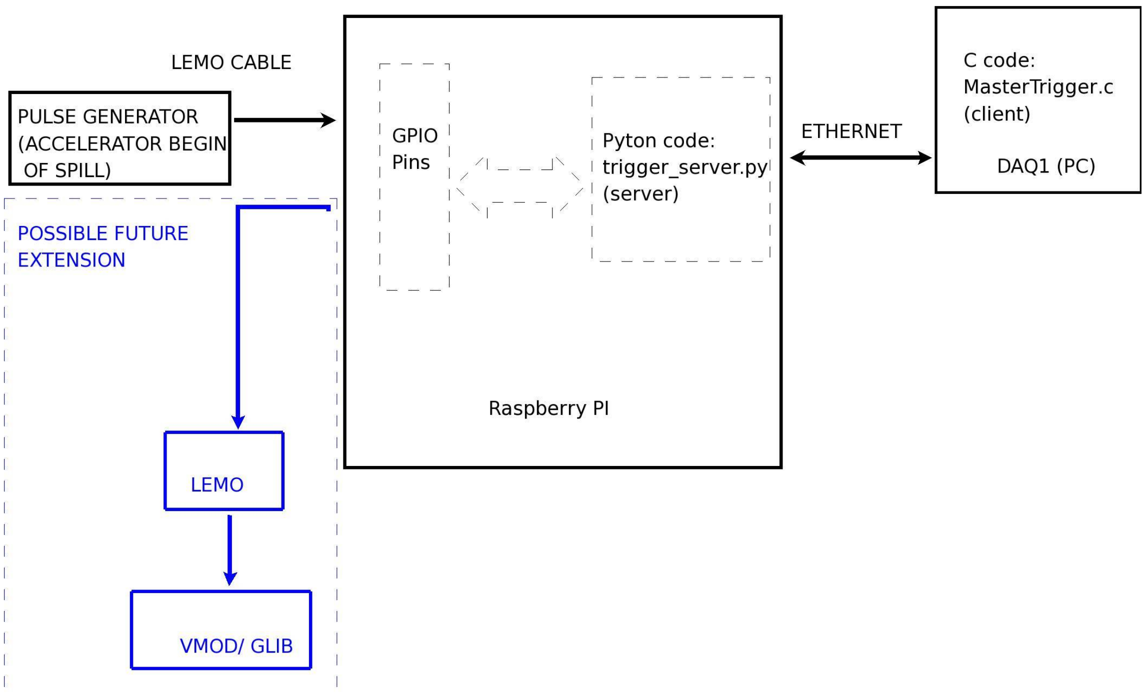
Gleb Lukicov (UCL)
g.lukicov@ucl.ac.uk
31 July 2014

Contents:

Introduction:

The purpose of this part of the experiment is to act a trigger for data acquisition equipment. This solution will be used for beam testing events.
The Trigger Server (Raspberry Pi) receives external Begin Of Spill (BOS) and End Of Spill (EOS) signals, and it commands the MasterTrigger frontend (FE) to start and stop. The MasteTrigger FE is then communicates with the Event Builder FE via the Midas solution.

The code is written in Python (see trigger_server.py) and is running on the Raspberry Pi. This way, Raspberry Pi acts as a server, with Master Trigger as a client. When an external signal is detected on the pin of Raspberry Pi a TCP message is sent to the Master Trigger via Ethernet, as shown the diagram below:


LEMO CABLE
PULSE GENERATOR (ACCELERATOR BEGIN OF SPILL)
GPIO Pins
Pyton code: trigger_server.py (server)
Raspberry PI
ETHERNET
C code: MasterTrigger.c (client)
DAQ1 (PC)
POSSIBLE FUTURE EXTENSION
LEMO
VMOD/ GLIB

iagram 1. The above diagram shows the concept as of 30 June 2014.

The **possible** future extension involves generating internal (to Raspberry Pi) EOS and BOS signals for VMOD/GLIB board.

A. Technical Specifications

Raspberry Pi:
Model B Revision **2.0 000e (R2)**

Chip	Broadcom BCM2835 SoC
CPU	700 MHz Low Power ARM1176JZ-F
GPU	Dual Core VideoCore IV
RAM	512MB SDRAM
Operating System	Linux (Raspbian OS v 3.6.11)
Physical Dimensions	8.6cm x 5.4cm x 1.7cm

Please note:

The current limit on a GPIO (General-purpose input/output) pin is 16mA, while the total current limit for all GPIO pins is **51mA** [2]. Care must be taken not to draw to much current from the Raspberry Pi. **The Raspberry Pi operates at 3.3 V** [2].

Technical Documentation is available here:
http://www.element14.com/community/docs/DOC-42993/l/raspberry-pi-single-board-computer#anchor3

UCL Specific Information:

Login Details for Raspberry Pi (rp0)
User: pi
Password: ucl@hep
Hostname: rp0.hep.ucl.ac.uk

Hwaddr: b8:27:eb:b4:87:07 (MAC address)
IP (LAB Network/ Labgate): 192.168.200.50
IP (HEP Network): 192.168.4.41

B. Qucik Set-up Guide for Trigger Server (Raspberry Pi) solution

*Raspberry Pi, comes with Raspian Linux OS v 3.6.11 (pre-installed on the SD card), which has Python pre-installed: v2.6, **v2.7 (default python)**, and v3.4. **The version 2.6 or 2.7 should be used**, because of compatibility with RPIO module and API buffer library (see later).*

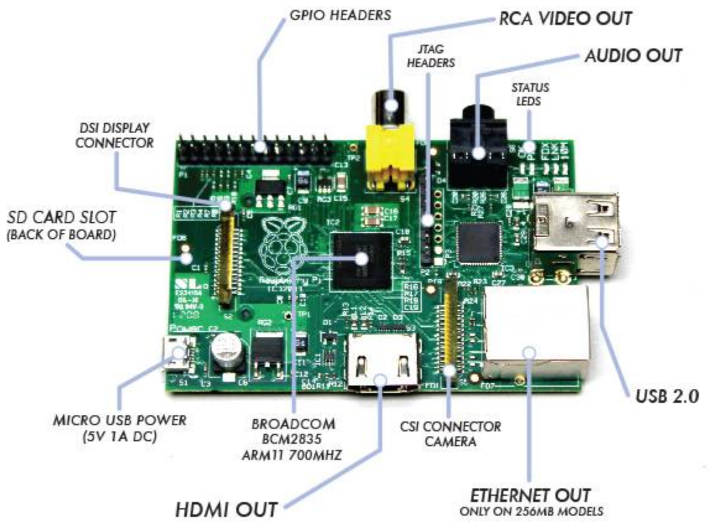

GPIO HEADERS
RCA VIDEO OUT
JTAG HEADERS
AUDIO OUT
STATUS LEDS
DSI DISPLAY CONNECTOR
SD CARD SLOT (BACK OF BOARD)
USB 2.0
MICRO USB POWER (5V 1A DC)
BROADCOM BCM2835 ARM11 700MHZ
CSI CONNECTOR CAMERA
HDMI OUT
ETHERNET OUT ONLY ON 256MB MODELS

Diagram 2. The image above shows Raspberry Pi with all the components indicated

Using Raspberry Pi is like operating a normal linux machine. **Keyboard, Mouse, Monitor, and Internet Connection are rquuired for initial set-up**.

Here is a generic Start Guide:

1. First of all, slot the SD card into the SD slot.
2. Plug in external mouse and keyboard to USB ports
3. Plug in a monitor into the HDMI output
4. Connect the Ethernet cable
5. Plug in the Micro USB Power – this will switch on the Raspberry Pi.

6. Now, follow the first-time boot installation: **create a username and a password**.
7. Make sure that the ssh is enabled. You can aslo change your timezone and location here. sudo raspi-config command will open this (boot) configuration tool again, if need arise later.
8. Now, set up the internet connection depending on your network settings: command `ifconfig` can be used to find the MAC address (HWaddr), `ifup etho, ifdown eth0` are aslo useful.

More detailed network set-up is available here:
https://learn.adafruit.com/downloads/pdf/adafruits-raspberry-pi-lesson-3-network-setup.pdf
and here: http://www.mathworks.co.uk/help/simulink/ug/getting-the-raspberry_pi-ip-address.html

Alternative start guide is available here: http://www.raspberrypi.org/wp-content/uploads/2012/12/quick-start-guide-v1.1.pdf

Once the network connection is established, it is possible (and highly recommended) to ssh into raspberry pi from more powerful machine. This is more convenint and time efficient. After log-in, the graphical system can be initiated by `startx` command, as an alternative to the terminal. Use `ssh –X` to employ graphical interface on a remote machine.

Experiment Specific Set-up:

This is a CONCISE quick guide. For detailed explanations please see the rest of this document.

1. Install RPIO module for Python v2.7 on Raspberry Pi:
Internet connection required.

- `$ sudo apt-get install python-setuptools`
  # Allows easy_install to be used

- `$ sudo easy_install-2.7 -U RPIO`
  # Installation of RPIO explicitly for Python v2.7

2. Copy the **trigger_server.py** onto Raspberry Pi:

It is important to run the trigger_server.py file using: `sudo` **(for RPIO utilisation) command. Also, python v2.6 or v2.7 must be used which are supported by RPIO (Python v 3+ is also supported BUT v3+ of Python gives an error like:** ***"Giving TypeError: Type str doesn't support the buffer API" for trigger_server.py)***

3. Attach the input from the signal generator (wave-form generator) to **GPIO 27** on Raspberry Pi (**see Pin Layout on p.6.**). This is important, as only this pin is **specified** in the trigger_server.py script.
Use the following setting on the signal generator:
Mode: Pulse. V=3.3V (amplitude), Period=10 s, Width=100 μs.

Detailed Instructions:

I. Pin Layout.

The diagram below shows the pin layout for Raspberry Pi Revision 2. The pins are referred by their BOARD number (1, 2, 3....26). However, in this document the convention is taken to label the pins in the script using BCM (Broadcom SOC channel) number. So that, GPIO 27 pin has BCM number 27, and BOARD number 13.
More information on pin functions in section 4 (p. 5).

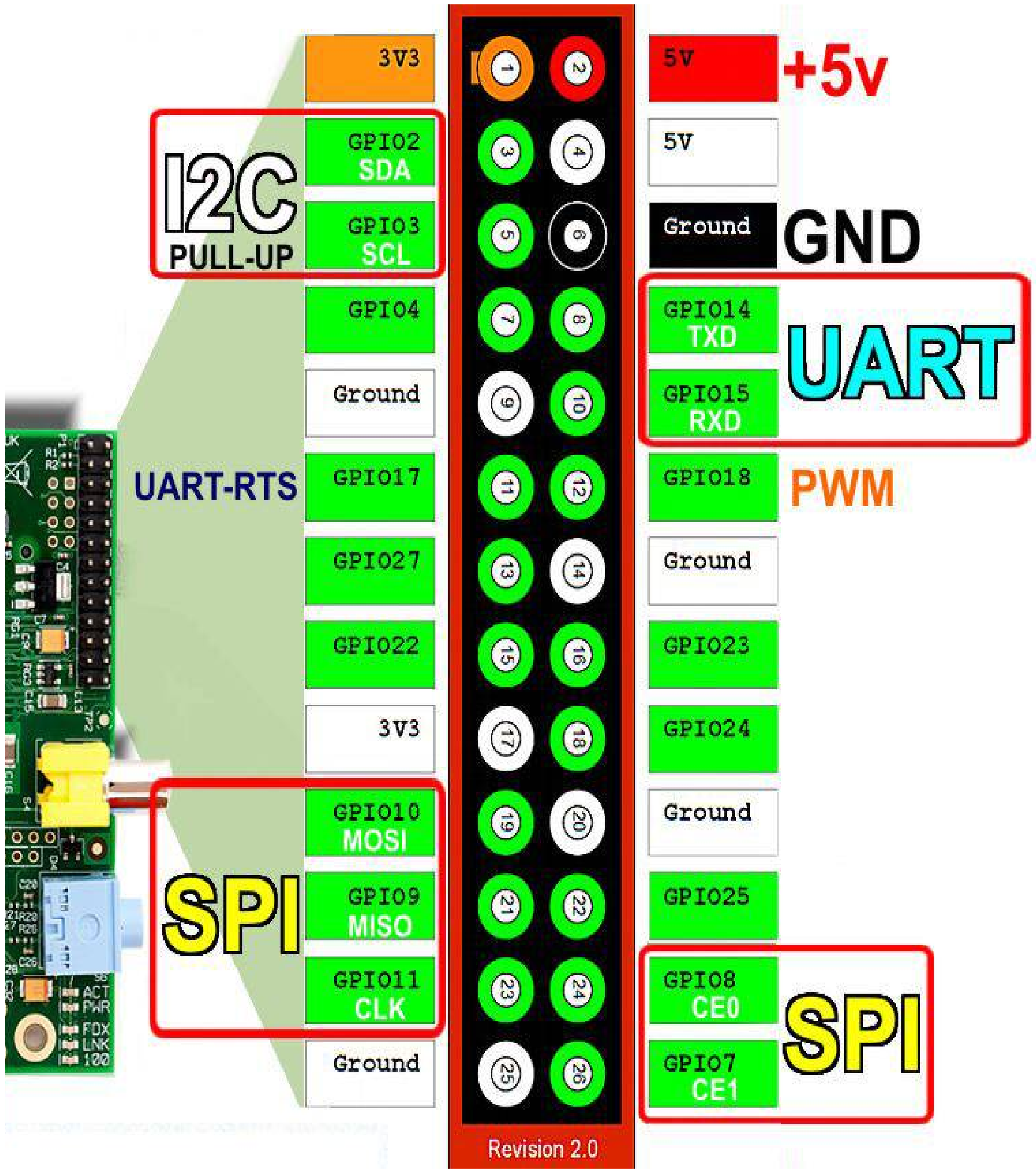


Diagram 3: Pin layout for Raspberry Pi (R2) Image source: Developer Blog [1]

II. Special Pin Functions [3]

Refer to diagram 3.

- **BCM_GPIO 17, 18*, 27, 22, 23, 24, 25: These are safe to use at any time and can be set to input or output with or without the internal pull-up or pull-down resistors enabled.**
- BCM_GPIO 4: It is used by the 1-Wire kernel driver and it can optionally be connected to a GPIO clock source.
- *BCM_GPIPO 18: If audio socket is not used at all, or the audio is going via the HDMI cable, then this pin is free to be used in PWM mode.
- BCM_GPIO 2 and 3: These are the I2C pins. They have on-board 1.8KΩ resistors.
- BCM_GPIO 8, 7, 10, 9 and 11: These are used for the SPI interface. These pins do not have any external pull up (or pull down) resistors.
- BCM_GPIO 14 and 15: These are used by the UART for Tx and Rx respectively.

Refer to diagram 3.

For our purpose any of the pins 17, 18*, 27, 22, 23, 24, 25 can be used.

III. Installing PRIO module for Python

RPIO (Raspberry Pi Input/Output) module is a powerful module for python written in C (see source code: https://github.com/metachris/RPIO) that uses DMA (Direct Memory Access) to generate output (e.g. PWM), or listen on inputs for one of the 18 GPIO pins on Raspberry Pi. Advantage of DMA over software is the reduced load on the CPU, which can affect timing critical events.

More information: https://pypi.python.org/pypi/RPIO

Installation:
Internet connection required.

```
$ sudo apt-get install python-setuptools
```
Allows easy_install to be used

```
$ sudo easy_install-2.7 -U RPIO
```
Installation of RPIO explicitly for Python v2.7

IV. Referencing GPIO pins

To execute Python command or script involving the use of GPIO the *sudo* command must be invoked.

To reference the relevant GPIO pin (see Diagram 1) its corresponding GPIO number - BCM - (e.g. 22 for “GPIO 22”) must be used in the script:

Example 1: In this simple script the GPIO22 (BOARD number 18) will be set at +3.3V.

```
Import RPIO # importing the RPIO library

RPIO.setmode(PIO.BCM) # setting pin referencing mode to BCM

RPIO.setup(22, GPIO.OUT) # setting GPIO22 pin as output

RPIO.output(22, True) # setting the logic on the pin as “ON”
```

The *GPIO.cleanup()* command should be used in the end to reset the status of any GPIO pins.

V. Limitations

The current limit on a GPIO pin is **16mA,** while the total current limit for all GPIO pins is **51mA** [2]. Care must be taken not to draw to much current from the Raspberry Pi. The Raspberry Pi operates at **3.3 V** [2].
When implementing external components (e.g. a switch) pull-up/pull-down resistors can be specified in the code:
For example, this will set pull down resistor and pull up resistor at pins 23 and 24, respectively:

```
RPIO.setup(23, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

RPIO.setup(24, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

****This section is for reference only:****
****Internal Trigger is no longer employed in the solution***

VI. Internal Trigger: Generating a pulse

Here is an example script for generating PWM from GPIO pins:

`from RPIO import PWM` # Importing PWM facility from RPIO module

`servo=PWM.Servo()` # Creating a PWM servo

`servo.set_servo(18, 200)` # Setting servo on GPIO18 (BCM) to 200 microseconds

`servo.set_servo(23, 200)` # Setting servo on GPIO23 (BCM) to 200 microseconds

To run the script:

sudo python test1.py

When the oscilloscope is connected to pins 18 and 23: the following could be seen on the screen:

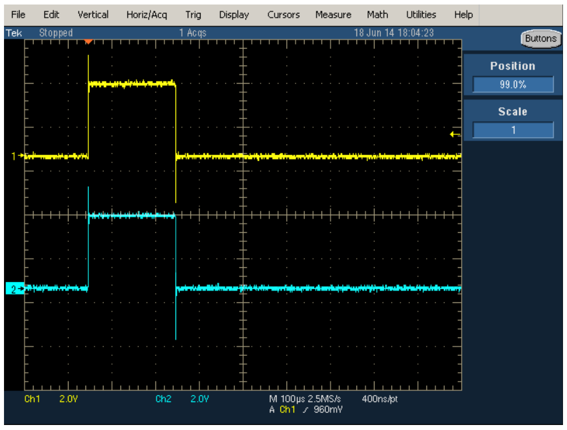
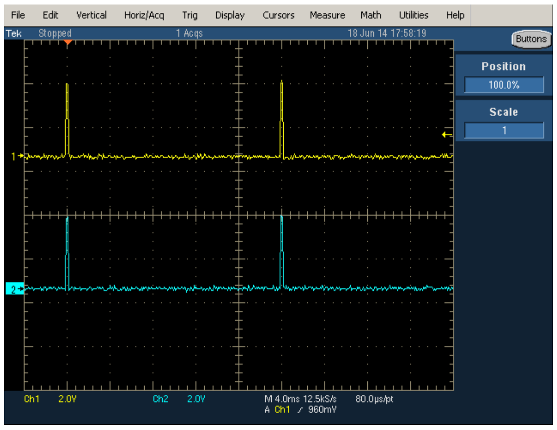

File Edit Vertical Horiz/Acq Trig Display Cursors Measure Math Utilities Help
Tek Stopped 1 Acqs 18 Jun 14 18:04:23
Buttons
Position 99.0%
Scale 1
Ch1 2.0V Ch2 2.0V M 100µs 2.5MS/s 400ns/pt
A Ch1 ⁄ 960mV

File Edit Vertical Horiz/Acq Trig Display Cursors Measure Math Utilities Help
Tek Stopped 1 Acqs 18 Jun 14 17:58:19
Buttons
Position 100.0%
Scale 1
Ch1 2.0V Ch2 2.0V M 4.0ms 12.5kS/s 80.0µs/pt
A Ch1 ⁄ 960mV

Diagram 4. The oscilloscope output. Timebase=100 µs, Amplitude=2V. The Channel 1 (GPIO 18) is indicated in yellow and Channel 2 (GPIO 23) is in blue. The leftmost image shows the pulse width, 100 µs, while the other image shows the period – 20 ms).

****This section is for reference only:****
****Internal Trigger is no longer employed in the solution***

VII. External Trigger: Listening on inputs

RPIO module has an inbuilt interrupt function called `RPIO.add_interrupt_callback`, which is ideal for our purpose to wait for EOS/BOS signals.

Interrupt function arguments:

1) GPIO 27 pin – specify which pin to use (BCM number)
2) callback defined by gpio_callback – function is called when interrupt on the pin occurs
3) edge: trigger on 'both' edges (other options: 'rising' or 'falling'
4) pull_up_down: Possible values are (i) pull UP resistor RPIO.PUD_UP
(ii) pull DOWN resistor RPIO.PUD_DOWN
(iii) no resistor on the pin selected* RPIO.PUD_OFF
*The resistor should be included in the circuit (pin input) by software for protection of Raspberry Pi.
5)threaded_callback: (i) True: the callback will be started inside a thread
(ii) False: the callback will block the execution of the code from executing until the next pin interrupt (i.e. no further callbacks dispatched in the meantime)
6) debounce_timeout_ms: if set, interrupt callback will not be started until specified time (in ms) have passed since the last interrupt.

RPIO.add_interrupt_callback(27, send_triggers, edge='both', pull_up_down=RPIO.PUD_UP, threaded_callback=False, debounce_timeout_ms=debounce_timeout)

VIII. References:

[1] Werner Ziegelwanger. Developer Blog. *GPIO layout.* Online. Available at: http://developer-blog.net/hardware/raspberry-pi-gpio-schnittstelle-teil-1. Created: 1 October 2013. Accessed: 16 June 2014.

[2] The Box. *Understanding Outputs.* Online. Available at: http://www.thebox.myzen.co.uk/Raspberry/Understanding_Outputs.html Accessed: 16 June 2014.

[3] Wiring Pi. *Special Pin Functions.* Online. Available at: http://wiringpi.com/pins/special-pin-functions Accessed: 17 June 2014